A Particle Swarm Optimization with Stagnation Detection and Dispersion

Chukiat Worasucheep

Abstract—**Particles or candidate solutions in the standard Particle Swarm Optimization (PSO) algorithms often face the problems of being trapped into local optima. To solve such a problem, this paper proposes a modified PSO algorithm with the stagnation detection and dispersion (PSO-DD) mechanism, which can detect a probable stagnation and is able to disperse particles. This mechanism will be described and its performance is evaluated using eight well-known 30-dimensional benchmark functions that are widely used in literature. The results show a promising alternative path for solving the common problem of local optima in PSO algorithms.**

I. Introduction

PARTICLE Swarm Optimization (PSO) is a population-based evolutionary computation technique first introduced in 1995 by Eberhart and Kennedy [1]. PSO algorithm is a nature-based stochastic optimization technique that simulates the social behavior of a group of simple individuals called *swarm* of *particles*. Each particle represents a potential solution, which is a point in a multi-dimensional search space. The particle has a fitness value and a velocity to adjust its flying direction in accordance to the best experience of the swarm and itself in order to search for a global optimum. PSO has become popular in the research field of optimization over the last decade for its simplicity of concept and implementation, few parameters and high convergence rate. Several reports indicate that PSO has a good performance in solving nonlinear optimization problems [2]-[7]. However the standard versions of PSO face a common problem of easily being trapped into a local optimum when solving complex multimodal problems [5]-[7]. In order to solve such a problem, this paper presents the Particle Swarm Optimization with stagnation detection and dispersion (PSO-DD) that uses a simple method to improve performance on complex problems.

This paper is organized as follows. Section II introduces the original Particle Swarm Optimization and some of its current variants. Section III describes the proposed Particle Swarm Optimization with Stagnation Detection and Dispersion (PSO-DD) algorithm. Section IV reports its performance evaluation using 8 well-known benchmark functions with the results and discussions. Section V concludes this paper with some future direction.

C. Worasucheep is with Faculty of Science, King Mongkut's University of Technology Thonburi, Tungkru, Bangkok 10140, Thailand (e-mail: chukiatw@yahoo.com).

II. Particle Swarm Optimization Algorithms

Particle Swarm Optimization (PSO) algorithm involves "flying" a swarm (or population) of particles through a problem space, in search of a single optimum or multiple optima. Each particle represents a possible solution to the optimization problem. In case of a basic PSO, each particle has its own velocity, a memory of the best position it has ever obtained (referred to as its *personal best* position), and knowledge of the best solution found by its neighborhood (referred to as the *global best* solution).

An original version of PSO introduced by its originators [2] consists of a group or swarm of N particles. The position of particle i at iteration t is represented by $\vec{x}_i(t) = (x_{i1}, x_{i2}, ..., x_{iD})$ and its velocity is represented by $\vec{v}_i(t) = (v_{i1}, v_{i2}, ..., v_{iD})$, where D is dimension of the optimization problem. Each particle adjusts its position in a direction toward its own personal best position and the global best position. For particle i, its velocity is calculated using

$$v_{ij}(t+1) = w \cdot v_{ij}(t) + c_1 \cdot r_{1j}(t) \cdot (pbest_{ij}(t) - x_{ij}(t)) + c_2 \cdot r_{2j}(t) \cdot (gbest_{ij}(t) - x_{ij}(t)) \tag{1}$$

which is then used to change its position according to

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{2}$$

where *pbest* and *gbest* are the personal best and global best positions, respectively, of the particle i. Subscript j indicates dimension and $j \in \{1,\ldots,D\}$. Parameters c_1 and c_2, termed as cognitive and social components, are acceleration coefficients that pull the particle toward personal best and global best positions. r_1 and r_2 are uniform random values between 0 and 1. w is the inertia weight which balances the exploration and exploitation abilities of the particles and results in fewer iterations on average in search of an optimal solution. The adjustment of velocity and position for each particle will be repeated until some stop criteria are met. Eberhart and Shi later introduced a time decreasing inertia factor to balance the global search and the local search abilities of the swarm [2]. Many researchers have worked on improving the swarm's performance in various ways, resulting in a number of variants such as those with fuzzy adjusted inertia factor [3], constriction factor [4], unification of global version and local version [5], or comprehensive learning capability for solving multi-objective optimization problem [6], etc.

One problem commonly found in standard PSO algorithms is the premature convergence in many optimization problems. According to equation (1), each particle is mainly influenced by its previous best position and the position of the global best particle. Once the global best particle gets stuck in a local optimum, every remaining particle will quickly converge to that position. Their updated velocities will also approach zero; and thus they lose exploration capability and stop moving finally. This stagnation causes particles not to arrive at the global or better optimum [7]-[9].

A few researches have been conducted to tackle this problem and employ some forms of mutation operators [10]-[15]. Higashi and Iba [11] use a mutation operator that changes a particle dimension value using a random number drawn from a Gaussian distribution like that used in Evolution Strategies. Esquivel and Coello [12] use a non-uniform mutation operator in which the mutation rate decreases during an optimization run. Ratnaweera *et al.* [13] implement a mutation operator to the particle velocities instead of the particle positions and the mutations are only applied to particles when the *gbest* solution has not changed for a number of algorithm iterations, i.e. the population has stagnated and converged on a minimum. If this is a local minimum, then the mutations should help particles accelerate out of it. Wang *et al.*'s method [14] employs opposition-based learning for each particle and applies a dynamic Cauchy mutation on the best particle to avoid premature convergence.

III. Particle Swarm Optimization with Stagnation Detection and Dispersion

Particle Swarm Optimization with Stagnation Detection and Dispersion (PSO-DD) is based on the standard PSO described earlier with some checking routine for local optimum. The details of PSO-DD algorithm for a minimization problem are as follows.

1. The process of velocity and position updates in each iteration works as in equations (1)-(2) in Section II. The cognitive and social acceleration coefficients, c_1 and c_2, are set constant at 1.49618. The inertia weight, w, is linearly decreased from 0.9 to 0.5 during the run.
2. The velocity of particle is very important for searching the optimum. In order to avoid wandering around the basin, the magnitude of velocity is limited to one quarter of the search range in each dimension. Similarly, the search region for each dimension is constrained when updating the position of a particle as in equation (2). If a particle i happens to fly beyond the boundary [Low_j, Hi_j] of dimension j, its position is bounced back half way of the velocity using

$$x_{ij}(t+1) = x_{ij}(t) - 1.5 \cdot v_{ij}(t+1) \qquad (3)$$

3. After the first quarter of all allowed generations, a convergence check is performed every 50 generation in order to avoid getting trapped into a local optimum. The simple check and fix routine works as follows:

   Let f_{best} be the function value of the best particle,
   f_c be the current f_{best},
   f_p be the previously recorded f_{best},
   v_c be the current average velocity of all particles,
   v_p be the previously recorded average velocity.

$$R = \left| \frac{1 - f_c / f_p}{1 - v_c / v_p} \right| \qquad (4)$$

   Here the previously recorded f_{best} and average velocity are from the prior 500 generations. R is the ratio of an improvement of f_{best} over an improvement of average velocity of all particles. Generally, in case of minimization problems, f_{best} is expected to decrease. During an optimization run, when stagnation is impending, the particles tend to move slower with their average velocity approaching zero. Moreover, f_{best} is hardly improved, the term f_c / f_p is approaching 1. Therefore, the ratio R will drop dramatically. If R falls below a certain threshold, then stagnation is anticipated.
4. When R is smaller than 10^{-5}, two actions will be taken. First, the inertia weight is reset half way back to its original value of 0.9, i.e. $w_{new} = (w + 0.9) / 2$. Second, for all particles except the best one, its velocity and position will be dispersed, with a random chance of 90%, as follows:
   4.1. Its velocity is reversed and multiplied by 100; i.e. $v_{ij} = -100.0 * v_{ij}$.
   4.2. Its position is offset with a randomized displacement of ±0.1% of the range in that dimension; i.e.

```
Offset = $rand(0.0, 1.0) * 0.001 * ( Hi_j - Low_j )$
If $rand(0.0, 1.0) < 0.5$ Then  $x_{ij} = x_{ij} - Offset$
                          Else  $x_{ij} = x_{ij} + Offset$
```

This dispersion method should help particles escaping from a local optimum.

IV. Performance Evaluation

Eight well-known benchmark functions used in [14], [16]-[17] are selected in this performance evaluation. They are minimization problems with high dimension; of which F1 to F4 are unimodal and F5 to F8 are multimodal. All 8 functions with their respective ranges and $f_{\min}$ are shown in Table I.

TABLE I
SEARCH RANGES AND GLOBAL OPTIMUM OF THE TEST FUNCTIONS

Function	Search Range	$f_{\min}$
$F1(x)=\sum_{i=1}^{n}x_i^2$	[-5.12, 5.12]	0
$F2(x)=\sum_{i=1}^{n}i\cdot x_i^2$	[-5.12, 5.12]	0
$F3(x)=\sum_{i=1}^{n}i\cdot x_i^4+random[0,1)$	[-1.28, 1.28]	0
$F4(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(1-x_i)^2\right]$	[-30, 30]	0
$F5(x)=\sum_{i=1}^{n}-x_i\cdot\sin(-\sqrt{\lvert x_i\rvert})$	[-500, 500]	-12569.5
$F6(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	[-5.12, 5.12]	0
$F7(x)=-20\cdot\exp\left(-0.2\cdot\sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n}x_i^2}\right)$ $-\exp\left(\frac{1}{n}\cdot\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	[-32, 32]	0
$F8(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	[-600, 600]	0

A. Experimental Setup and Criteria

All benchmark functions are tested for 30 dimensions with the maximum number of function calls (MaxNFC) allowed at 100,000. For each function, 50 independent runs with the different random seeds are performed, and the fitness of the best particles (f_{best}) throughout each optimization run is recorded and reported. The results below 10^{-25} are reported as 0. The optimization program was written in C and tested on Intel Core2Duo Pentium® 2.0 GHz with 1 GB of main memory running Windows Vista Home Premium.

The mean, standard deviation, the best (lowest), and the worst (highest) values of resulted fitness values from 50 runs are calculated for each benchmark function. These basic statistical results are then compared to those from standard PSO (PSO) and Opposition-based PSO with Cauchy Mutation (OPSO) as shown in Table II. For a fair comparison, PSO-DD uses the same number of particles (N = 10) and the maximum number of function calls (MaxNFC = 100,000) as in OPSO and PSO algorithms. Note that OPSO sets parameters as follows: $c_1 = c_2 = 1.49618$, $w = 0.72984$, and Vmax set to half range of the search space. Details of OPSO algorithm can be found in [14].

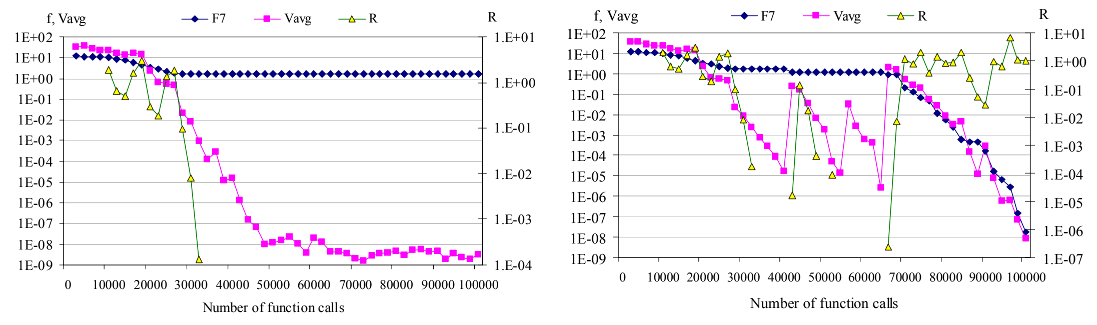
B. Comparisons Results

Table II reports the results achieved for functions F1 to F8 using the proposed PSO with Stagnation Detection and Dispersion (PSO-DD) in comparison to those from OPSO and standard PSO (PSO) obtained from [14]. The results shown in boldface are the best ones among all three algorithms. Except for functions F1 and F2, both PSO-DD and OPSO result in f_{best} below 10^{-25}. The following outcomes can be observed:

1. The sphere functions (F1 and F2) are very simple for both OPSO and PSO-DD algorithms. In fact, both algorithms achieve f_{best} below 10^{-40} for the best runs.
2. For function F3, which includes noise, PSO-DD achieves marginally better results than OPSO, and both algorithms are superior to standard PSO.

TABLE II
THE RESULTS ACHIEVED FOR F1 TO F8 USING DIFFERENT ALGORITHMS. THE RESULT BELOW 10^{-25} ARE REPORTED AS 0.

Function	PSO-DD		OPSO		PSO	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
	Best	Worst	Best	Worst	Best	Worst
F1	0	0	0	0	6.06E-07	2.07E-06
	0	0	0	0	1.98E-19	8.95E-06
F2	0	0	0	0	3.50E-05	1.60E-04
	0	0	0	0	1.26E-18	8.66E-04
F3	**1.26E-02**	**6.68E-03**	1.84E-02	5.36E-03	9.75E-02	7.03E-02
	3.72E-03	**4.17E-02**	1.14E-02	3.29E-02	1.33E-02	2.82E-01
F4	34.1207	29.3705	**0**	**0**	2.87	6.31
	3.49E-06	76.8321	**0**	**0**	3.02E-04	21.46
F5	-10712.9	408.7	**-10986.7**	**207.344**	-6705.10	631.20
	-11731.4	-9983.6	-11250.6	**-10674.5**	-8621.4	-5303.60
F6	**31.702**	**11.754**	49.950	11.290	62.220	12.700
	11.848	**80.591**	36.810	59.700	34.820	83.580
F7	**6.41E-08**	**4.40E-07**	1.190	1.060	7.490	2.080
	2.80E-14	**3.14E-06**	0	3.350	4.170	12.830
F8	**0.028**	**0.029**	0.047	0.043	0.659	1.15E-04
	0	**0.127**	0	0.176	0.896	3.820



(a) PSO without Stagnation Detection and Dispersion (b) PSO-DD

Fig. 1. f_{best}, $Vavg$ and R for function F7.

3. For F4 or Rosenbrock's function, which is non-separable and multi-modal, PSO-DD's results are worse than the other two algorithms because the function landscape has very narrow valley from local optimum to global optimum. If PSO-DD is equipped with some local search or a higher magnitude of dispersion, it could provide a better result. This should be further investigated.
4. For F5 or Schwefel's function with deep local optima being far from the global optima, the mean result of PSO-DD is marginally inferior to that of OPSO, but superior to standard PSO. Notice however that the best result of PSO-DD (-11731.4) is better than that of OPSO.
5. For functions F6 to F8, PSO-DD is superior to both OPSO and PSO with the statistical significance at 95% confidence interval. Especially for F7 or Ackley's function which has one narrow global optimum basin and many minor local optima, PSO-DD's particles successfully fly to the global optimum basin while the other two algorithms are trapped in some local optima.

C. *Effects of Swarm Dispersion*

Fig. 1 compares the progressed f_{best}, the average velocity of all particles (V_{avg}) and the ratio R during a run of function F7. The comparison is between PSO-DD without stagnation detection and dispersion in figure 1-(a) and PSO-DD algorithm in figure 1-(b). It can be observed that, during the early iterations, the f_{best} improves slowly due to a large number of local optima. Average velocity (V_{avg}) starts quickly falling after 27,000 NFC and thus makes the ratio R dive. This shows a sign of stagnation, implying the particles get trapped in some local optimum. In figure 1-a), f_{best} does not improve once NFC is approximate 30,000 meaning that the particles get trapped in some local optima at f_{best} is at 1.65.

In contrary, in figure 1-b) once the ratio drops below the threshold (10^{-5}) at approximate 38,000 NFC, the PSO-DD disperses the particles. This results in a higher average velocity (V_{avg}) or v_c in equation (4), and higher ratio R. Note however that the dispersion at this time cannot sufficiently lead most particles to the right path. The same phenomenon repeats again at approximate 55,000 NFC. Finally, at approximate 65,000 NFC, the algorithm detects stagnation and successfully disperses enough particles toward the path leading to be near optima. Therefore, f_{best} can further decrease and finally finish at $8.54\text{x}10^{-9}$ when NFC = 100,000. Note that the discontinuity of R line in the figure is because the unchanged f_{best} makes R approach zero and thus R can not be shown in the logarithmic scale.

D. *Effects of Swarm Size*

In the performance evaluation using eight benchmark functions described earlier, the swarm size or the number of particles is fixed at ten. It is interesting to investigate the effects of swarm size on outcomes whether it can improve f_{best} within the same MaxNFC at 100,000. This subsection runs the same set of benchmark functions again but with the swarm size varied from 10 to 70 at a step size of 10. The means, standard deviations, the best, and the worst f_{best} are recorded and reported in Table III. Figure 2 illustrates the minimum f_{best} of running each function with the varying swarm size. According to the table and the figure, the following observations can be obtained:

1. For simple functions like sphere functions as in F1-F3, the optimal swarm size is around 30-40 particles for 30 dimensional problems. Too few particles may not be sufficient for a convergence. However, too many particles may be wasteful and also delay the convergence.
2. For more complex multimodal functions, a smaller swarm size results in a better convergence except F8 (Griewank's). The linkage among variables in F8's product of cosine term makes it difficult to reach the global optimum especially with a smaller swarm size.

TABLE III
THE RESULTS ACHIEVED USING DIFFERENT SWARM SIZE.

	Swarm size	Mean	S.D.	Best	Worst		Swarm size	Mean	S.D.	Best	Worst
F1	10	6.79E-30	3.37E-29	5.65E-42	2.28E-28	F5	10	-10712.90	408.70	-11731.40	-9983.58
	20	1.75E-47	8.99E-47	1.82E-66	5.73E-46		20	-10729.00	349.34	-11503.50	-10082.20
	30	3.11E-58	1.33E-57	1.19E-69	9.30E-57		30	-10647.50	349.56	-11385.10	-9963.80
	40	9.15E-55	4.39E-54	3.38E-62	3.00E-53		40	-10525.90	309.27	-11266.70	-9877.22
	50	7.08E-50	3.27E-49	5.64E-55	2.26E-48		50	-10511.40	435.72	-11858.90	-9569.00
	60	1.46E-45	7.83E-45	9.53E-51	5.51E-44		60	-10451.90	379.97	-11148.20	-9588.74
	70	1.26E-41	8.15E-41	7.12E-46	5.82E-40		70	-10295.10	402.36	-11286.40	-9391.34
F2	10	2.29E-30	1.58E-29	8.37E-45	1.13E-28	F6	10	31.7021	11.7540	11.8475	80.5914
	20	9.24E-54	6.17E-53	8.80E-68	4.41E-52		20	43.1331	32.0287	17.0700	223.4370
	30	7.15E-58	4.05E-57	6.11E-71	2.81E-56		30	40.6283	12.7305	22.8840	74.6218
	40	3.06E-57	1.77E-56	5.29E-66	1.26E-55		40	40.3555	11.1722	20.8941	72.6318
	50	2.57E-51	1.10E-50	4.43E-60	7.57E-50		50	42.8903	26.1090	14.9244	206.3380
	60	7.45E-47	2.83E-46	1.04E-51	1.87E-45		60	37.1318	10.6965	12.9345	65.6672
	70	2.64E-42	1.11E-41	1.97E-46	7.58E-41		70	35.7389	9.0423	14.9244	51.7378
F3	10	1.26E-02	6.68E-03	3.72E-03	4.17E-02	F7	10	6.41E-08	4.40E-07	2.80E-14	3.14E-06
	20	8.40E-03	4.03E-03	2.42E-03	2.17E-02		20	4.96E-02	0.24501	1.38E-14	1.50175
	30	6.89E-03	3.22E-03	1.37E-03	1.65E-02		30	1.02E-01	0.35443	2.09E-14	1.64622
	40	5.78E-03	2.33E-03	2.18E-03	1.27E-02		40	3.28E-01	0.59907	1.38E-14	1.77800
	50	6.11E-03	2.50E-03	2.04E-03	1.57E-02		50	8.37E-02	0.33163	1.38E-14	1.50175
	60	6.47E-03	2.28E-03	1.93E-03	1.19E-02		60	1.18E-01	0.35855	1.38E-14	1.34042
	70	5.72E-03	2.03E-03	2.08E-03	1.12E-02		70	6.48E-02	0.25791	1.38E-14	1.15515
F4	10	34.1207	29.3705	3.49E-06	76.8321	F8	10	0.02837	0.02921	0	0.12691
	20	31.0228	32.3494	5.65E-11	134.022		20	0.01875	0.02084	0	0.08523
	30	29.8205	27.3051	2.79E-08	76.2956		30	0.01546	0.01978	0	0.09775
	40	30.0726	27.8804	0.0238119	80.5028		40	0.01500	0.01485	0	0.05857
	50	34.1949	27.904	2.09E-06	80.3869		50	0.01544	0.01955	0	0.10289
	60	31.1824	29.2679	3.55E-06	132.671		60	0.01191	0.01195	0	0.05137
	70	36.1586	30.2049	0.0001906	133.672		70	0.01042	0.01310	0	0.05133

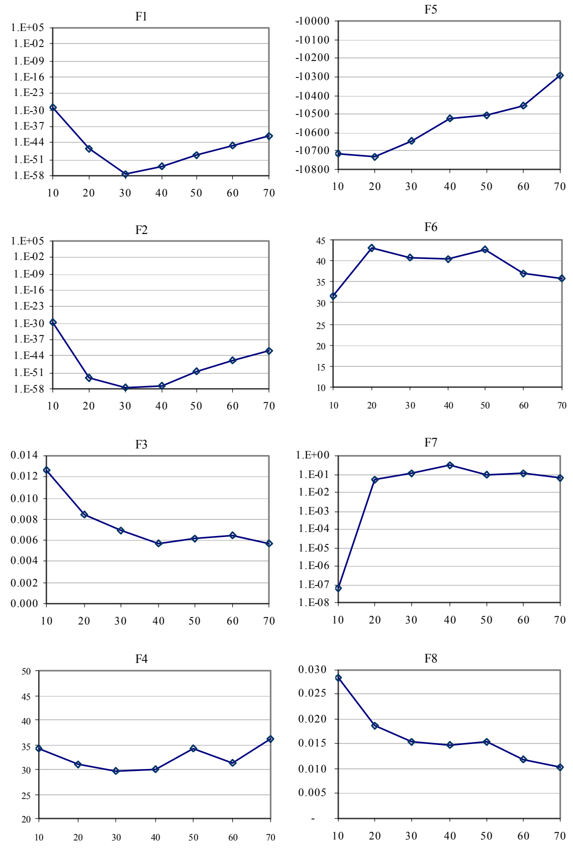


Fig. 2. Plots of f_{best} with varying swarm sizes for F1-F8 within 100,000 NFC.

V. CONCLUSIONS AND FUTURE WORKS

This paper proposes the PSO-DD algorithm that has a mechanism to solve the common problem of being trapped into local optima in complex functions. This mechanism checks a ratio of the improvement in the current best function value to the average velocity of particles. If the ratio falls below a certain threshold, the algorithm will disperse the particles for further exploration. The performance of the algorithm is evaluated using 8 well-known benchmark functions. The results show a superior performance over some other recent PSO variants in most functions. Some future works should focus on incorporating local search and comprehensive study of sensitivity analysis of the new internal parameters.

ACKNOWLEDGMENT

The author would like to thank his wife for her patience in proof reading and three anonymous referees for their constructive suggestions to this work.

REFERENCES

[1] J. Kennedy, R.C. Eberhart, "Particle swarm optimization", *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, 1995, pp. 1942-1948.
[2] Y. Shi, R. C. Eberhart, "A modified particle swarm optimizer", *Proceedings of the IEEE Congress on Evolutionary Computation*, 1998, pp. 69-73.
[3] Y. Shi, R. C. Eberhart, "Particle swarm optimization with fuzzy adaptive inertia weight", in *Proc. Workshop Particle Swarm Optimization*, Indianapolis, IN, 2001, pp. 101-106.
[4] M. Clerk and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.* Vol. 6, no. 1, Feb. 2002, pp. 58-73.
[5] K. E. Parsopoulos and M. N. Vrahatis, "UPSO—A unified particle swarm optimization scheme," in *Lecture series on computational science*, 2004, pp. 868-873.

[6] V. J. Huang, Suganthan, P. N., and J. J. Liang, “Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems”, *Int. J. of Intelligent Systems*, vol. 21, no. 2, 2006, pp. 209-226.

[7] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm Intelligence*, San Francisco: Morgan Kaufmann Publishers, 2001.

[8] W. J. Zhang and X. F. Xie, “EPSO: Hybrid Particle Swarm with Differential Evolution Operator", in Proc. of IEEE SMC, WA, USA, 2003, pp. 3816-3821.

[9] F. V. Bergh, "An Analysis of Particle Swarm Optimizers", PhD thesis, University of Pretoria, 2002.

[10] F. V. Bergh, A. P. Engelbrecht, “Cooperative Learning in Neural Networks using Particle Swarm Optimization”, *South African Computer Journal*, No. 2000, pp. 84-90.

[11] N. Higashi and H. Iba, “Particle swarm optimization with Gaussian mutation,” in *Proceedings of the IEEE Swarm Intelligence Symphosium 2003*. IEEE Press, 2003, pp. 72–79.

[12] S. C. Esquivel and C. A. Coello Coello, ‘‘On the use of particle swarm optimization with multimodal functions,’’ in *Proceedings of the 2003 Congress on Evolutionary Computation*. IEEE Press, 2003, pp. 1130– 1136.

[13] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, ‘‘Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients,’’ *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[14] H. Wang, Y. Liu, S. Zeng, H. Li, and C. Li, “Opposition-based Particle Swarm Algorithm with Cauchy Mutation”, *Proc. of the 2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 4750-4756.

[15] X. Wang, Y. Wang, H. Zeng, and H. Zhou, “Particle Swarm Optimization with Escape Velocity”, 2006, pp. 457-46.

[16] X. Yao, Y. Liu and G. Lin, “Evolutionary Programming Made Faster”, *IEEE Trans. on Evol. Comp*., vol. 3, July 1999, pp 82-102.

[17] K. Veeramachaneni, T. Peram, C. Mohan, L. A. Osadci, “Optimization Using Particle Swarms with Near Neighbor Interactions”, *Proc. Genetic and Evolutionary Computation (GECCO 2003)*, 2003, pp. 110-121.